Proof that NP equals P

© Chinmoy Mukherjee 2025-2045, all rights reserved, no part of the document can be reproduced without written permission from author

Proof that NP equals P
Introduction
    Author reference:
    NP=P is proved!
Flowersweeper: The Game
    Game Board & Objective
Proof of NP-Completeness
    Flowersweeper is NP-Hard
        Wires and Logic Gates
        The Full Construction
Conclusion

Introduction

Author reference:

https://chinmoymukherjee.com/ ,
https://www.linkedin.com/in/gcparchitect

NP=P is proved!

I have designed a new NP-Complete problem "Flowersweeper" in line with "Minesweeper".
Where pests are placed in petals of the flower.

For any caterpillar who wants to drink honey stored at the center of the flower.
This is an NP-Complete problem. Proof is below.
But when the caterpillar transforms into butterflies, it can see the center of the flower and fly directly to center of the flower and drink honey in O(N) time!

For the caterpillar, drinking honey from the center of flower, laden with pests is a NP-Complete problem.
For the butterfly, drinking honey from the center of flower is solvable in O(n) time.
Which proves NP-Complete problem can be solved in polynomial time.

Grok

Flowersweeper: The Game

Flowersweeper is a logic puzzle game played on a flower with a unique structure. The goal is to help a friendly caterpillar find a safe path from the outer edge to the center of the flower, avoiding hidden "pest" caterpillars.

Game Board & Objective

The Flower: The game board is a flower consisting of N concentric circles of petals. Each petal is a single cell.

Pests: Some petals hide pest caterpillars (equivalent to mines in Minesweeper).

The Goal: The player's objective is to find and secure a continuous path of safe (pest-free) petals for a caterpillar to travel from any petal on the outermost circle to any petal in the innermost circle.

Gameplay: The player clicks on petals to clear them.

If a cleared petal contains a **pest**, the player loses.

If a cleared petal is **safe**, it reveals a number. This number indicates the exact count of pests hidden in all adjacent petals. (Adjacency includes petals in the same circle and connecting petals in the circles immediately towards and away from the center).

Using these numbers as clues, the player must deduce which un-cleared petals are safe to form a path to the center.

Proof of NP-Completeness

To prove that Flowersweeper is **NP-Complete**, we must satisfy two conditions:

The game is in the complexity class **NP**.

The game is **NP-Hard**.

Flowersweeper is in NP

A problem is in **NP** (Nondeterministic Polynomial time) if a potential solution can be verified for correctness in polynomial time.

Let's define the Flowersweeper decision problem as: "Given a partially revealed Flowersweeper board, does there exist a consistent assignment of pests to the remaining petals that permits a safe path to the center?"

To prove this is in NP, consider a proposed solution, which we'll call a "certificate". This certificate would be a complete map showing the locations of all pests on the un-cleared petals.

A verification algorithm can check this certificate in polynomial time (an amount of time that scales reasonably with the size of the flower board) by performing two simple checks:

Clue Consistency Check: For every numbered petal on the board, the verifier counts the number of adjacent pests in the proposed solution map. If this count matches the number on the petal for all revealed petals, the map is consistent. This check is fast and scales linearly with the number of revealed petals.

Path Existence Check: If the map is consistent, the verifier performs a search (like a Breadth-First Search or DFS) on the flower graph to see if a path of safe petals exists from the outer circle to the inner circle. This search also runs in polynomial time.

Since a proposed solution can be fully verified in polynomial time, **Flowersweeper is in NP**. ✓

Flowersweeper is NP-Hard

A problem is **NP-Hard** if it is at least as "hard" as any other problem in NP. We prove this by taking a known NP-Complete problem and "reducing" it to Flowersweeper. This means we show how any instance of the known hard problem can be converted into a Flowersweeper board. If we can solve the Flowersweeper board, we can solve the original problem.

We will reduce the **Boolean Circuit Satisfiability (Circuit-SAT)** problem, a classic NP-Complete problem, to Flowersweeper.

Circuit-SAT Problem: Given a Boolean logic circuit made of AND, OR, and NOT gates, is there an assignment of True/False values to the inputs that makes the final output of the circuit True?

The reduction involves constructing "gadgets" on the Flowersweeper board using petals and number clues that mimic the behavior of a logic circuit.

Wires and Logic Gates

We can build wires and logic gates by placing numbered clues in a way that constrains the possible locations of pests in adjacent un-revealed petals. A "signal" in a wire can be represented by two possible pest arrangements.

Wire Gadget: A wire can be made from a chain of un-revealed petals. Clues are placed to force an alternating state. For example, a 2 clue placed adjacent to only two petals, A and B, forces one to be a pest and one to be safe. By chaining these, we can propagate a signal. State(A) -> !State(B) -> State(C). This allows a signal to travel across the board.

NOT Gate: A NOT gate is simply one segment of the alternating wire described above. If the input is petal A, the

output is petal B, and the state of B is always the opposite of A.

OR Gate: We can construct an OR gate ($C=A \vee B$) using a central clue that links three wires. For example, a petal with a '2' clue can be made adjacent to the output petals (hA ,hB ,hC) of three wires. This constraint forces that exactly two of these petals must contain pests. By carefully setting up the wires, this configuration correctly mimics the truth table of an OR gate.

If inputs A and B are False, their wire outputs (hA ,hB) will be pests. To satisfy the '2' clue, the output wire for C (hC) must be safe, making C False.

If either A or B is True, its respective wire output is safe. To satisfy the '2' clue, C's output (hC) must be a pest, making C True.

Since we can build NOT and OR gates, we can also build AND gates (since $A \wedge B = \neg(\neg A \vee \neg B)$) and thus any logic circuit.

The Full Construction

For any given Circuit-SAT problem, we construct a corresponding Flowersweeper board. We build gadgets for each input, wire, and gate in the circuit.

The final output wire of the circuit is linked to a special petal, let's call it the **gate petal**.

We create a long, otherwise obvious path for the caterpillar, starting from the outer edge and ending at a petal right next to the **gate petal**.

A final '1' clue is placed adjacent to only these two petals: the last petal of the path and the **gate petal**.

This setup means the path is safe **if and only if** the **gate petal** contains a pest. We design our circuit gadgets such that the **gate**

petal containing a pest corresponds to the circuit's output being True.
Therefore, a safe path for the caterpillar exists if and only if there is an input assignment that makes the circuit's output True. Solving the Flowersweeper board is equivalent to solving the Circuit-SAT problem.
Since we can convert any Circuit-SAT problem into a Flowersweeper board in polynomial time, **Flowersweeper is NP-Hard**. ✓

Conclusion

Because Flowersweeper is both in **NP** and is **NP-Hard**, it is proven to be **NP-Complete**. This means that, like Minesweeper, solving a Flowersweeper board in the general case is a computationally very hard problem.